# Iris Flower Classification

September 22, 2025

[4]:
```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
```

[5]:
```
#Data loading
df=pd.read_csv("Iris.csv")
print("Dataset loaded successfully")
df.head(5)
```

```
Dataset loaded successfully
```

[5]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

[6]:
```
#Basic EDA
print("Informatio of dataset")
df.info()
```

```
Informatio of dataset
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
```

```
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

[7]:
```
print("Missing values of dataset")
df.isnull().sum()
```

```
Missing values of dataset
```

[7]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

[8]:
```
print("Statistic of dataset")
df.describe()
```

```
Statistic of dataset
```

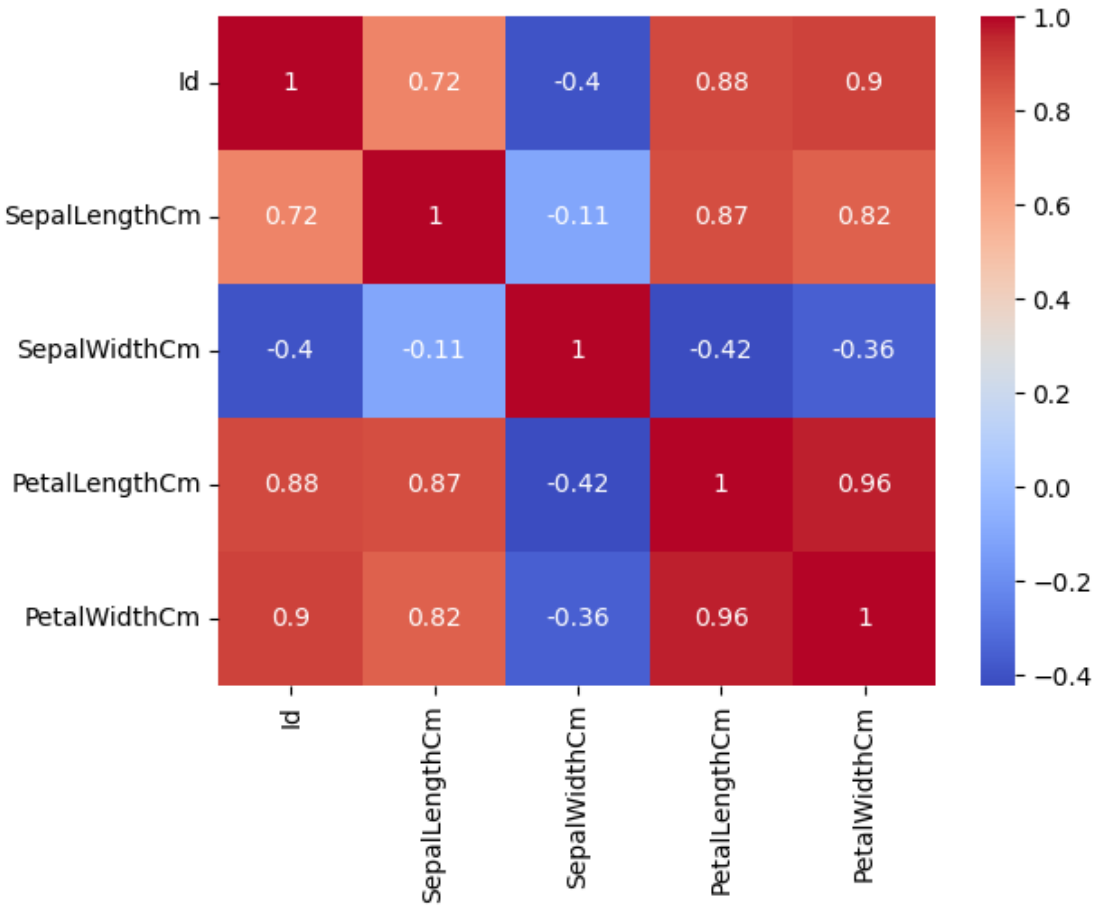
[8]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|---|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 75.500000 | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| std | 43.445368 | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| min | 1.000000 | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| 25% | 38.250000 | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| 50% | 75.500000 | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| 75% | 112.750000 | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| max | 150.000000 | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

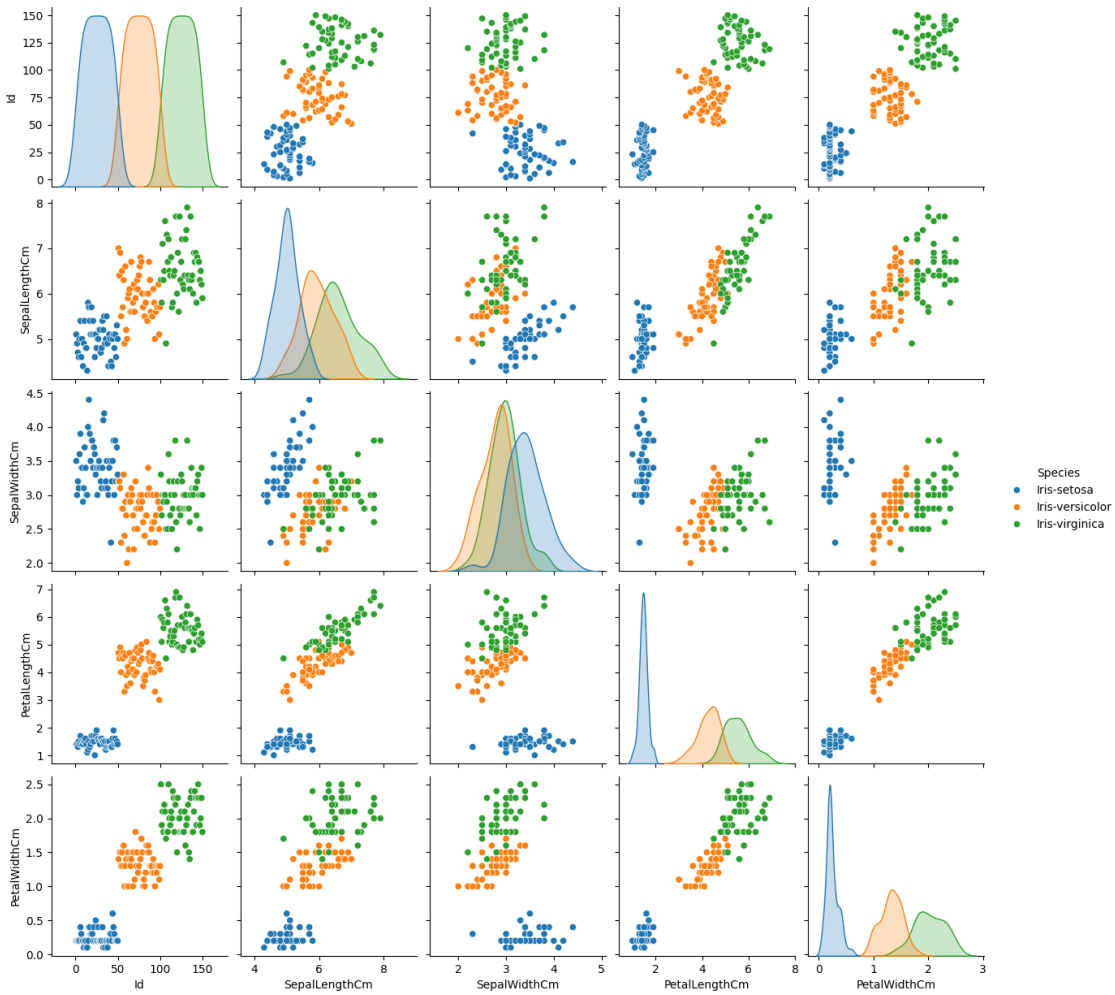
[9]:
```
print("Correlation Heatmap")
corr=df.corr(numeric_only=True)
sns.heatmap(corr,annot=True,cmap='coolwarm')
```

```
Correlation Heatmap
```

[9]:
```
<Axes: >
```




[10]:
```
sns.pairplot(df,hue='Species')
plt.show()
```




[11]:
```
#Feature Selection
X=df.drop("Species",axis=1)
y=df["Species"]
#Feature Scaling
scaler=StandardScaler()
scaler.fit_transform(X)
#Train-Test split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

[12]:
```
#Creation and evaluation of Model
#function to create and evaluate model
def modelselect(model,name):
    model.fit(X_train,y_train)
    y_prediction=model.predict(X_test)
    print(f"{name} Result")
#Accuracy prediction
```

```
    acc=accuracy_score(y_test,y_prediction)
    print("Accuracy Score is:",acc * 100,"%")
    print("--------------------------------------")
#Classification Report
    report=classification_report(y_test,y_prediction,zero_division=0)
    print("Classification Report:",report)
    print("----------------------------------------------")
#Confusion Matrix plotting
    matrix=confusion_matrix(y_test,y_prediction)
    sns.heatmap(matrix,annot=True,fmt='d',cmap='Blues')
    plt.title(f"Confusion Matrix:{name}")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
```

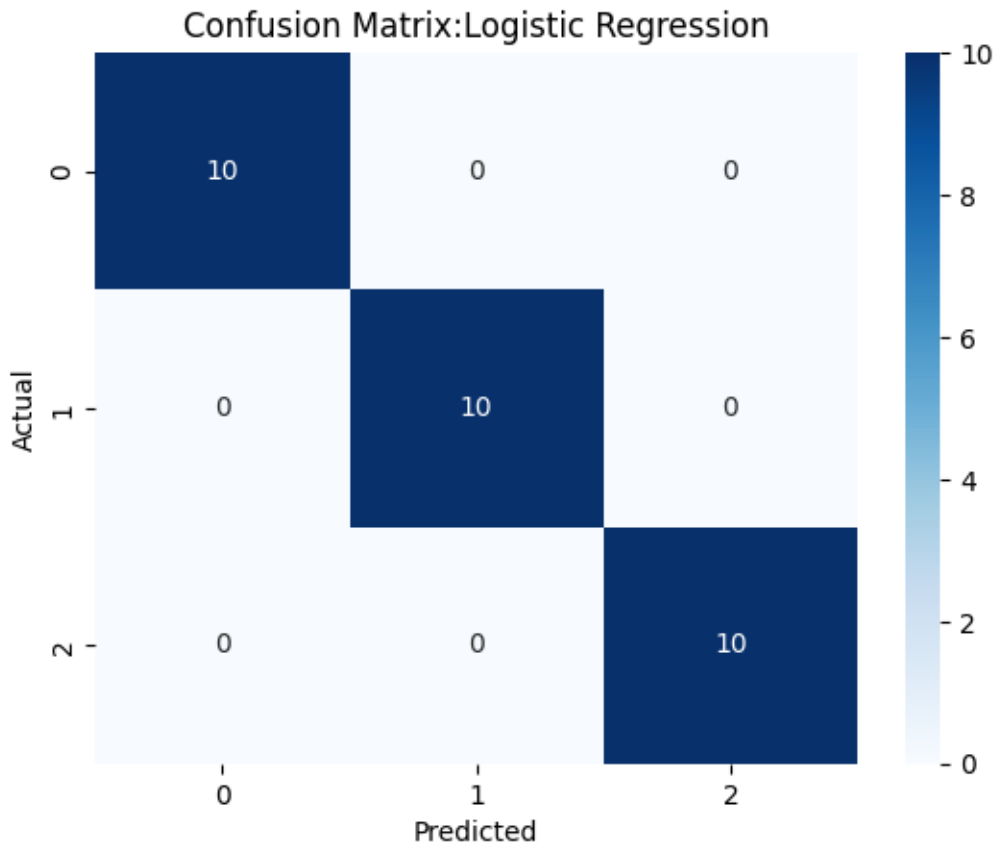
[13]:
```
modelselect(LogisticRegression(max_iter=1000),"Logistic Regression")
```

```
Logistic Regression Result
Accuracy Score is: 100.0 %
--------------------------------------
Classification Report:                    precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00        10
 Iris-virginica       1.00      1.00      1.00        10

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30

----------------------------------------------
```



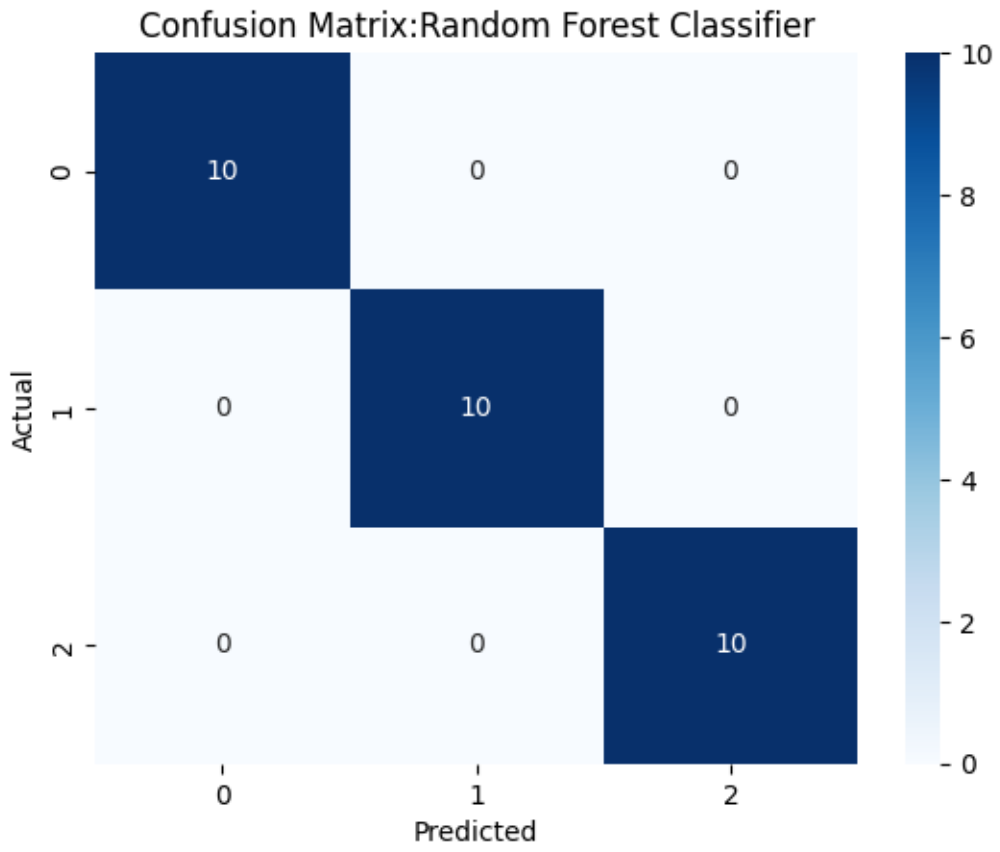

[14]:
```
modelselect(RandomForestClassifier(n_estimators=100,random_state=42),"Random␣
↪Forest Classifier")
```

```
Random Forest Classifier Result
Accuracy Score is: 100.0 %
-------------------------------------
Classification Report:                    precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00        10
 Iris-virginica       1.00      1.00      1.00        10

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30

-----------------------------------------------
```



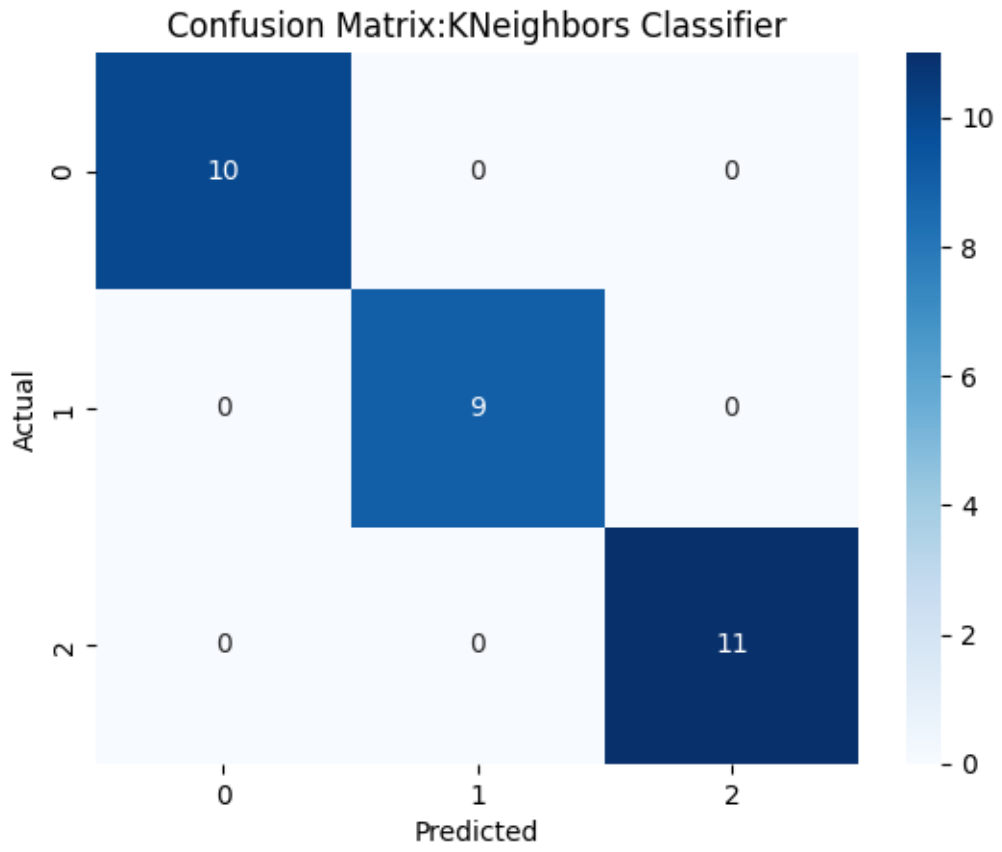

[34]:
```
modelselect(KNeighborsClassifier(n_neighbors=3),"KNeighbors Classifier")
```

```
KNeighbors Classifier Result
Accuracy Score is: 100.0 %
-------------------------------------
Classification Report:                    precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30

-----------------------------------------------
```




[42]:
```
import warnings
warnings.filterwarnings("ignore", message="X does not have valid feature names")
```

[43]:
```
import numpy as np
import pandas as pd
newdata = np.array([[1,5.3, 3.5, 1.4, 0.2],
                    [2,6.5, 3.0, 5.2, 2.0]])
data = pd.DataFrame(newdata, columns=X.columns)
print(data)
#Prediction for new data
scaled=scaler.fit_transform(data)

model = KNeighborsClassifier(n_neighbors=4)
model.fit(X_train,y_train)
prediction=model.predict(scaled)

for flower, pred in zip(newdata, prediction):
    print(f"Flower {flower} --> Predicted Species: {pred}")
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
```

```
0  1.0           5.3           3.5           1.4           0.2
1  2.0           6.5           3.0           5.2           2.0
Flower [1.  5.3 3.5 1.4 0.2] --> Predicted Species: Iris-setosa
Flower [2.  6.5 3.  5.2 2. ] --> Predicted Species: Iris-setosa
```